GraphQL

Expliqué par

David Gilson

https://github.com/gilsdav

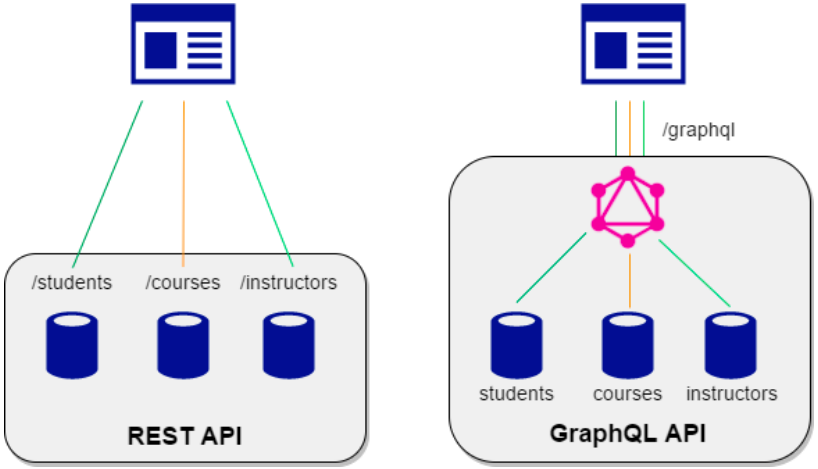
Est-ce répandu ?



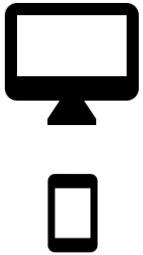


GraphQL

- A query language for your API
- Fini de faire plusieurs APIs pour la même ressource
- Front-end agnostic



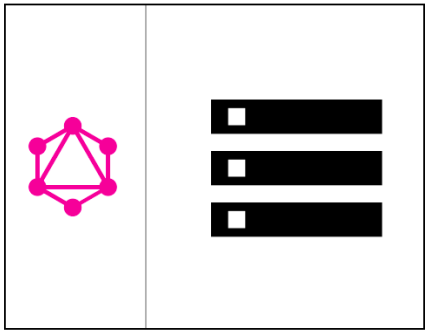

1
HTTP GET
{
"user": {
"id": "er3tg439frjw"
"name": "Mary",
"address": { … },
"birthday": "July 26, 1982"
}
}
/users/<id>
/users/<id>/posts
/users/<id>/followers
2
HTTP GET
{
"posts": [{
"id": "ncwon3ce89hs"
"title": "Learn GraphQL today",
"content": "Lorem ipsum … ",
"comments": […],
}]
}
/users/<id>
/users/<id>/posts
/users/<id>/followers
3
{
"followers": [{
"id": "leo83h2dojsu"
"name": "John",
"address": { … },
"birthday": "July 26, 1982"
},
…]
}
HTTP GET
/users/<id>
/users/<id>/posts
/users/<id>/followers



```
query {
  User(id: “er3tg439frjw”) {
    name
    posts {
      title
    }
    followers(last: 3) {
      name
    }
  }
}
```

HTTP POST

```
{
  “data”: {
    “User”: {
      “name”: “Mary”,
      “posts”: [
        { title: “Learn GraphQL today” }
      ],
      “followers”: [
        { name: “John” },
        { name: “Alice” },
        { name: “Sarah” },
      ]
    }
  }
}
```



Protocols

- HTTP
- WebSocket

- Aucun problème de compatibilité

Glossaire

- **Query**
  - Récupération (HTTP - POST)
- **Mutation**
  - Création (HTTP - POST)
  - Modification (HTTP - POST)
- **Subscription**
  - Écoute d'un évènement (WebSocket)
  - Récupération de données en temps réel (WebSocket)

GraphQL en Angular

- Apollo
- https://www.apollographql.com/docs/angular/

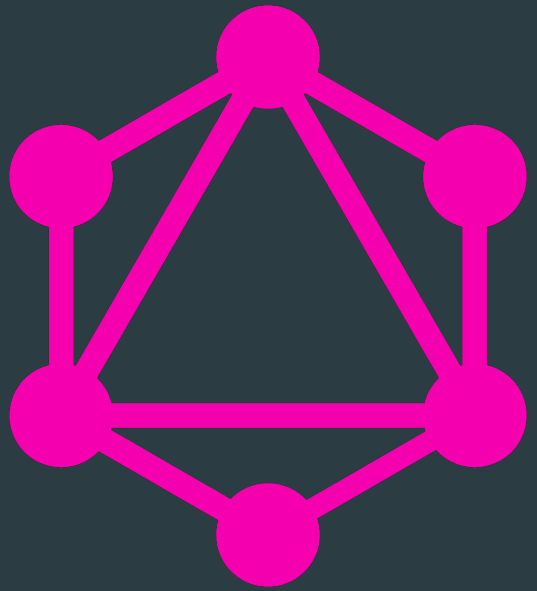
- Modules
  - apollo-angular
    - apollo-client
      - graphql
      - graphql-tag
  - apollo-angular-link-http
    - apollo-link (http)
    - Apollo-link-ws (ws)

Cas d’étude

- Application de création de pizza basée sur une API GraphQL
  - https://github.com/gilsdav/angular-graphql-workshop
    - https://github.com/gilsdav/dotnet-graphql-workshop

- Afficher les pizzas
- Créer/mettre à jour une pizza
- Écouter un évènement à la création/modification d’une pizza

- Accéder à une API protégée (session via cookie)



Query

Query

- Apollo = 2 types de query
  - Query
    - Requête unique (équivalent du GET http)
    - apollo.query
  - WatchQuery
    - Requête émise à un intervalle régulier
    - pollInterval: 5000
    - apollo.watchQuery

Query

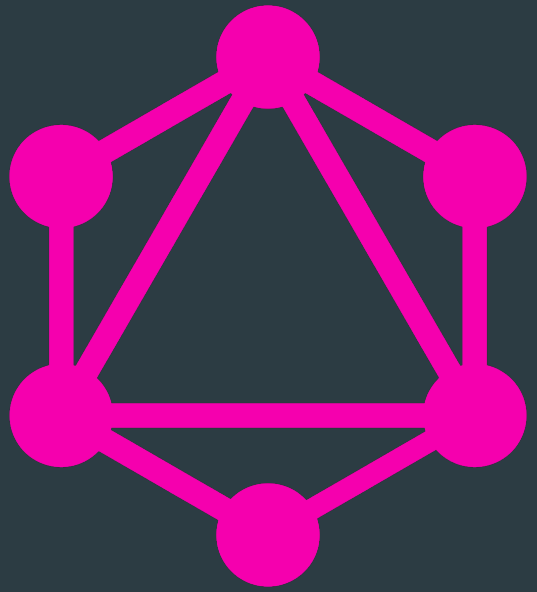
- Utilisation de « graphql-tag » pour écrire les queries

```
const query = gql`
   query GetAllPizzas {
      pizzas {
         id,
         name,
         toppings {
            id,
            name
         }
      }
   }
`;
```

Query

- Query avec paramètre

```
const query = gql`
   query GetPizza($id: Int!) {
      pizza (id: $id) {
         id,
         name,
         toppings {
            id,
            name
         }
      }
   }
`;
return this.apollo.query({
   query,
   variables: {
      id
   }
}).pipe(map(result => result.data.pizza));
```

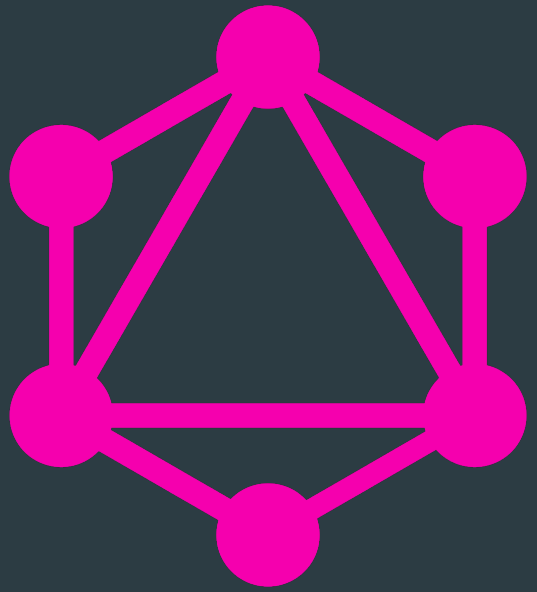


Mutation

Mutation

- Même chose que pour les queries

```
const mutation = gql`
   mutation UpdateePizza($id: Int!,$name: String!, $toppings: [Int]!) {
      updatePizza (pizza: {id: $id, name: $name, toppings: $toppings }) {
         id,
         name,
         toppings {
            id,
            name
         }
      }
   }
`;
return this.apollo.mutate<{ updatePizza: Pizza }>({
   mutation,
   variables: {
      id: payload.id,
      name: payload.name,
      toppings: payload.toppings.map(top => top.id)
   }
}).pipe(map(response => response.data.updatePizza));
```

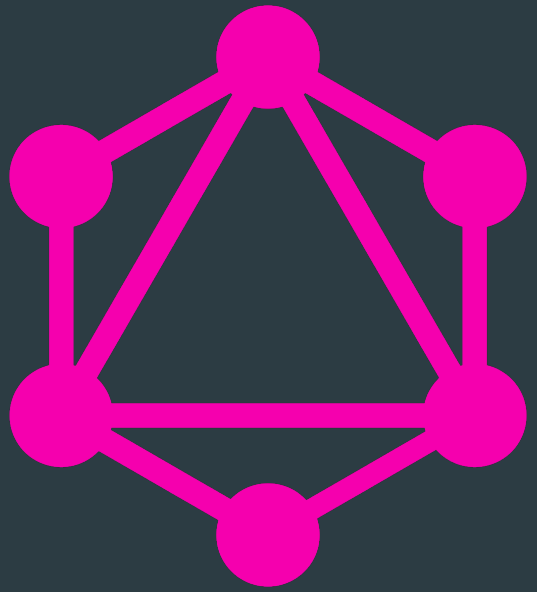


Subscription

Subscription

- Utilisation de la méthode « subscribe »

```
const query = gql`
   subscription PizzaCreation {
      pizzaAdded {
         id,
         name
      }
   }
`;
return this.apollo.subscribe<{ pizzaAdded: Pizza }>({
   query,
   fetchPolicy: 'no-cache'
}).pipe(
   filter(result => !!result.data), // only if value
   skip(1), // do not get repetition
   map(response => {
      return response.data.pizzaAdded;
   })
);
```



Configuration

Configuration

- Options de base

```
watchQuery: {
    fetchPolicy: 'cache-and-network',
    errorPolicy: 'ignore',
    pollInterval: 5000 // 0 = uniq call (same as "query")
},
    query: {
    fetchPolicy: 'no-cache',
    errorPolicy: 'all'
}
```

Configuration

- Création d'une api nommée

```
const options: Options = { uri: authUri, withCredentials: true };

apollo.createNamed('auth', {
   link: httpLink.create(options),
   cache: new InMemoryCache(),
   defaultOptions
});
```

Configuration

- Configuration websocket

```
const options: Options = { uri: mainUri, withCredentials: true };
const http = httpLink.create(options);

const ws = new WebSocketLink({
   uri: mainUriWs,
   options: {
      reconnect: true
   }
});

const link = split(({ query }) => {
   const { kind, operation }: any = getMainDefinition(query);
   return kind === 'OperationDefinition' && operation === 'subscription';
}, ws, http);
```

Références

- https://www.apollographql.com/docs/angular/
- https://medium.com/@snowronark/implementing-graphql-subscription-feature-with-angular-7-and-apollo-basic-2490bd4faab0
- https://graphql.org/
- https://github.com/gilsdav/dotnet-graphql-workshop
- https://github.com/gilsdav/angular-graphql-workshop